Mechatronic blockset for Simulink-concept and implementation

Ravn, Ole; Szymkat, Maciej; Uhl, T.; Betemps, M .; Pjertursson, Anders; Rod, J.

Published in:
Proceedings of the 1996 IEEE International Symposium on Computer-Aided Control System Design

Link to article, DOI:
10.1109/CACSD.1996.555348

Publication date:
1996

Document Version
Publisher's PDF, also known as Version of record

Proceedings of the 1996 IEEE International Symposium
on Computer-Aided Control System Design
Dearborn, MI • September 15-18, 1996

TP07 4:40

Mechatronic BlockSet for Simulink - concept and implementation

Ole Ravn[*], M. Szymkat[**], T Uhl[***], M. Betemps[****], A. Pjetursson[*] and J. Rod[**]

*Department of Automation, Technical University of Denmark,
Building 326, DK-2800 Lyngby, Denmark, E-Mail: or@iau.dtu.dk

**Division of Control Engineering, ***Dept. of Mechanical Eng. and Robotics,
St.Staszic Technical University, al.Mickiewicza 30, 30-059 Krakow, Poland, E-Mail: msz@earth.ia.agh.edu.pl

****Laboratoire d'Automatique Industrielle, Institut National des Sciences Appliquees
INSA-Batiment 303, 20 Avenue Albert Einstein, 69621 Villeurbanne, France E-Mail: mbetemps@insa.insa-lyon.fr

Abstract

The paper describes the design considerations for a system for modelling and simulation of mechatronic systems. The system is based on a component concept enabling the designer to pick component models that match the physical components of the setup to be modelled from a block library. Another important feature of the system is the ability to change the complexity of the simulation model in a simple, powerful and well structured way. This feature makes it simple for the designer to evaluate the influence of including different aspects of the components. The complexity can be changed both on the component level and for the whole model. This library that can be extended by the user contains standard components, such as DC-motors, potentiometers, encoders, pneumatic elements, and a Maple based facility to generate symbolic equations of motion. To evaluate the concepts the Mechatronic Simulink Library blockset has been implemented as a prototype based on MATLAB and Simulink and has been used to model several mechatronic systems. The library is presently being tested in different projects.

Keywords: Modelling, Mechatronics, CACSD, Simulation.

1 Introduction

Modelling is essential when engineers develop new systems. It is convenient to test the performance of the controller and the total system without having to construct and assemble it. There are at least two important groups of models to be considered in CACE. Models that are used for testing the system performance through analysis or simulations, and models that are used as a starting point for the design of the controller.

Normally the first kind is more complex and as close to the real system as possible while the second kind is simplified, containing only modelling features of the system that are important for the control. The second kind could be called design models, while the first kind could be denoted validation models.

Design models are sometimes derived by simplification of the validation models. The focus in this paper is on deriving validation models. We also restrict ourselves to computer based models, leaving out the mechanical small-scale models sometimes used.

Several ways of obtaining computer based models exist. In this work the traditionally used approach in mechanical engineering is taken. This approach is sometimes called deductive modelling. Based on the physical laws (Newton etc.) and the geometry of the system a set of differential equations is derived from the structure of the model. The parameters of the model can be derived in three ways:

- From data sheets available for the components of the system
- Measured by performing special experiments suited for finding the parameter in question
- Found using an optimization routine and data taken from a run of the real system. This is also called calibration of the model.

As opposed to this approach stochastic modelling could be used. This approach uses data-sets recorded from the real systems and based on these, using identification algorithms [6] an optimal model description is found including both structure and parameters.

In this paper the first approach is used. In order to simplify the modelling component libraries containing models of subsystems and components that can be reused are introduced.

Mechatronics is a combination of the words mechanics and electronics and the use of the term originated in Japan in the 1970's but it has since then become a standard term. The precise meaning of the term is not well defined as several interpretations exists [15]. There is a general consensus that mechatronics consists of the elements mechanics, electronics and software. In our approach the element 'control' has been added as shown in figure 1.

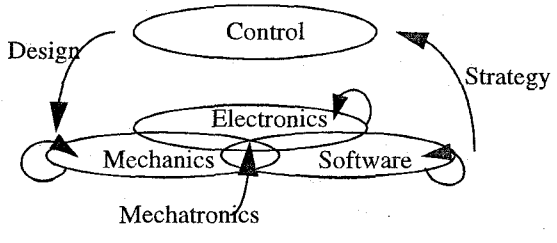

Figure 1. Mechatronics

Included in the term mechatronics are aspects from different fields of expertise and elements of 'concurrent engi-

0-7803-3032-3/96/$5.00 © 1996 IEEE

neering' or 'integrated design' are clearly found. A mechatronics approach to a construction task will deal with the problems encountered in a parallel fashion involving all the relevant fields of expertise and not as in more traditional construction first deal with the mechanical aspects, then add the electronics and lastly add the software and control algorithms.

To be able to work on a construction task in a mechatronic fashion it is essential that the tools and methods used supports all the involved fields of expertise or at least are open to integration with other methods and tools. One of the large problems encountered when solving a problem in a mechatronic way is the lack of tools and methods that support analysis, controller design and simulation in this way. The typical tools are emerging from one of the above mentioned domains of expertise as for instance the simulation tool SPICE from electronic circuits. Many of these tools support the inclusion of components from other domains like mechanical components in SPICE, but it is often difficult as the differential equations are the vehicle used in the inclusion. This generates a barrier for the designer as finding these equations demands a detailed knowledge of the other domains. A mechanical designer will for instance rarely have detailed knowledge about the modelling of PWM amplifiers.

The most laborious task in the initial phase of the analysis of a mechatronic robotic system is the derivation of dynamic models. Model equations may be derived using paper and pencil or this task may be supported by the computer. The first approach may become very time-consuming and always brings a risk of human errors. In such cases, computerized symbolic manipulation is clearly faster and safer. The purely numerical approach has many drawbacks e.g. round-off errors generated when evaluating expressions. Some combination of numerical and symbolic approach in analysis of dynamics of mechatronic system seems to be most appropriate.

As the input the definition of the topology of the system is required. The system is fully described by the geometry and physical data. In order to derive model equations some theoretical principles are usually employed. There exists two groups of formalism used to generate the equations of dynamics. The first is based on vectorial mechanics and includes the Newton-Euler approach and the second is attributed to analytical mechanics and encompasses different formulations (d'Alembert, Jourdain, Lagrange or Hamilton). Other principles are used sometimes such as Gibbs-Appell's and Gauss's, but they are not that popular as those of Euler or Lagrange.

The Symbolic Math Toolboxes for MATLAB brings a new dimension to this numerically oriented package. This encourages the construction of alternative solutions eliminating (in some areas) the need of using specialized software for multibody systems analysis, at least for relatively simple systems.

2 Mechatronic Simulink Library

The aim of the development of the Mechatronics Simulink Library (MSL) has been the make a domain independent library of component models to be used for modelling electromechanical systems. The designer should be able to use the blocks without a detailed domain specific knowledge and the resulting models should be well suited for solving control problems. The domain specific knowledge is embedded in the blocks and makes it useable for designers without much domain specific knowledge.

An important feature of MSL is that the user can choose the component models from a library. Typically the library will contain a number of generic component models and a number of models of the specific components that are available to the designer. These component models eliminates the need to model common components each time they are to be used in a system, likewise it is not necessary to find the values of parameter in data sheets, as the data for the different component types are build into the library. Furthermore the inclusion of new specific component models is easy using the generic component models.

Another important feature of MSL is the possibility of changing the complexity of the model in a simple fashion. Each component model reacts to the choice of different parameters that indicate the desired complexity of the component. The parameters can be set for the individual components and for the whole system model, which makes it very easy the evaluate the effect of a given change of complexity.

The following section describes how the concepts are build into the component model structure and following this a short summary of the some of the components of the prototype implementation is given. The current prototype version of MSL is implemented using MATLAB/ Simulink, as there was a strong desire to use a widely available environment and not to implement existing functionality in the project. The concepts however are independent of the implementation environment and could be realised using other packages. More detailed information on MSL can be found in [15], [14], [9], [10], [11], [12] [16] and [18].

3 Component structure

At the first glance Simulink seems like a system well suited for modelling continuous system. However if certain guidelines are not followed regarding the structure of the resulting models does not become very reusable. The problems faced are the choice of input/ output signals for the blocks, the choice of level of granularity and finally the way to specify parameters for a specific component.

The choice of granularity or complexity is important as too detailed models become very slow and hard to simulate. Using the complexity parameters of MSL the freedom exists to implement models at different levels of complexity concurrently. The idea behind MSL is to decompose the

system models into component models directly related to the physical setup of the system being modelled. A DC motor and an amplifier are built as two different component models and a current control of the motor is realised using a feedback of the armature current of the motor to the amplifier component. This is not an optimal approach from a pure simulation speed point of view but on the other hand the system model becomes simple to maintain and change, if for instance the effect of voltage control is to be evaluated. If the simulation performance problems become to overwhelming aggregated multi-components models can be realised to over come this solving the algebraic loops symbolically before simulation. Integrating the symbolic elimination of algebraic loops and the component based concept presented here is a very promissing future direction in standard environments, the current environments however does not support this.

It is also of importance to determine which inputs and outputs the components should have. [2]. For some types of components inputs and outputs are simple to choose, for instance in the case of the potentiometer where the input is the shaft position and the output is the voltage, the supply voltage can be input as a parameter. The problem is more difficult if a DC motor is considered where either the armature current or the voltage can be input and the output could be the shaft speed for a given load intertia and torque or the torque could be output. The importance of a structured approach to choosing the input, output and parameters is clear, because of the complications related to changing component models and connecting different system models. The choice of input and output signals for some components in the prototype implementation of MSL is shown in table 1.

Table 1. MSL input/outputs

Component	Input	**Output**
DC Motor	Load torque, Voltage	acc, speed, pos, current
Combined amplifier	Signal voltage, Tacho output, Load current	Voltage
Bridge amplifier	Signal voltage	Voltage
PWM amplifier	Voltage	Voltage
Ideal gear	acc, speed, pos, Load torque	acc, speed, pos, Load torque
Flexible arm	Rotation angle Angular acceleration	End point angle, Load torque, Strain, time functions

In the MSL system the parameters of a model are put into two different categories. One for parameters that are typically constant and well determined for the component type and one for the parameters that are less determined and that could be changed in the course of the use of the model. The border line between the two categories are not quite well defined, but it is rather easy to move a parameter between the two categories through a modification of component model. All data in the first category is collected in a database implemented as a m-function called `msldata`. Data for new component types and new components are simply added using a normal text editor to the m-file that makes up the database. The m-file is a function which is called with the component type and name, for instance

```
[Ra,L,Ke,Jm,Fluid,Coulomb,Stiction,Vmax]=
      msldata('DCmotor','ms012C')
```

and returns the parameters:

```
Ra = 3.7
L= 460e-6
Ke = 17.09e-3
Jm = 1.894e-6
Fluid = 0.0
Coulomb = 0.0
Stiction = 1.71e-3
Vmax = 12.0
```

These values are from the manufacturers data sheet.

The other type of parameters are transferred from the dialogue box of the masked block and can be input directly as a number or refer to a global variable in the MATLAB workspace. In table 2 is shown which parameters are put into what category in the prototype implementation.

Table 2. MSL parameters

Component	**Database**	**Dialogue**
DC Motor	Ra, L, Ke, Jm, Fluid, Coulomb, Stiction, Vmax	Name, Fluid Coulomb, Stiction
Combined amplifier	Vlim, Ilim, Bandwidth Rg, Vlim1, Vlim2	Name K1,.K2, K3
Bridge amplifier	K1, Vlim1, K2, Vlim2	Name
PWM amplifier	freq, amplitude	PWM Name
Ideal gear		N, Jg, Fluid
Flexible arm	Jb, alfa, Rw, Ww, Kmom, Phi2dd, PhiL	Name, Strain gauge pos, Number of modes

The dialogue box in general contains the name of the component type for use in the database. To make it possible for the designer to change the complexity of the system model in a simple way and asses the effect of the change of different parasitic effects on the results of the simulations five different general effects has been considered. They are `noise (N)`, `saturation (S)`, `friction (F)`, `quantisation (Q)` and `dynamics (D1/D2)`.

Apart from these switches the sampling interval is also a global variable, but if needed it is also possible to use different sampling intervals by introducing more global variables or inputting the sampling interval directly in the dialogue box.

The effect of the five general switches are translated for each component type into specific effects directly related to the component in question. See table 3. Using this mechanism it is simple to change the complexity of a complete system model inputting the name of global variables in the dialogue box (This is the default behaviour). By changing the global variables the complexity of the complete model can be changed. If the effect of a change of complexity of a single component is to be checked the desired value is input into the dialogue box of the component or an extra global variable can be used. In this way the user has total control over the model complexity in a simple fashion.

Table 3. MSL granularity

Component	N	F	S	Q	D1	D2
DC Motor		X[a]				X[b]
Combined amplifier			X[c]		X[d]	X[e]
Bridge amplifier			X[f]			
PWM amplifier						
DC Tacho	X[g]					
Encoder				X[h]		
Strain Gauge			X[i]			
Gear		X[j]				

a. Static, coulomb and fluid friction.
b. Induction
c. Output voltage saturation
d. Time delay in output
e. Current limitations
f. Output voltage saturation
g. Commutation noise
h. Digital quantization
i. Maximum voltage output of instrumentation amplifier
j. Fluid friction

4 Components

In the prototype implementation of MSL a number of components often used in the laboratory have been made. Other components can be added as needed, they could be hydraulic or pneumatic. In figure 2 the current groups of components are shown. It should be noted that MSL is designed in such a way that the addition of new component types is simple and the user should use this facility as only the most basic component types are included from the beginning.

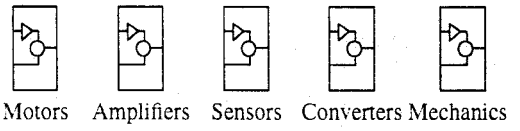

Figure 2. Mechatronic Simulink Library

Motors includes only DC motors but other drive types can be added by the user. In the figure the dialogue box is shown, in which the type of DC motor is input as well as if the parameters of friction, stiction and fluid friction should be taken from the database (default) or the dialogue box. Furthermore it can be chosen if friction should be simulated or not and the level of extra dynamics included in this case the electrical time constant of the drive. As also shown in the figure there are two variants of the drive, a normal and a triggered version. The triggered blocks is an undocumented feature of Simulink 1.3 enabling the user to only simulate certain blocks in the diagram. This is utilized in MSL to enhance simulation performance of the simplest complexity level significantly. In the traditional version all parts of a component are simulated even though the results from some blocks are not used. The whole model with full complexity is always simulated in the non-triggered version. The triggered version should always be used, the normal version is only included because of the undocumented nature of the triggered block facility that may be changed in future versions of Simulink.

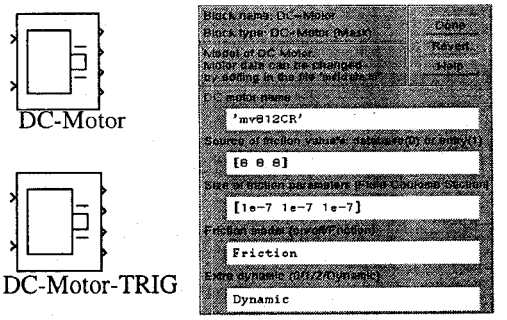

Figure 3. DC motor symbols and dialogue box.

The input variables are the load torque and voltage and the output variables are shaft position, speed, acceleration. The armature current is also seen as a output in order to be able to simulate a current limiter in the amplifier.

Amplifiers. In this group power amplifiers as well as signal, pre-amplifiers and bridge amplifiers for strain gauges are modelled. The load current is an input signal for power amplifiers along with voltage and is normally connected to the corresponding output signal of the DC-motor model. This enables the simulation of a current limiting behaviour.

Sensors. A number of control relevant sensor components are modelled, DC-tacho, potentiometer, strain gauge and encoder.

Converters. Models of the AD and DA converters with delay and quantisation.

Mechanical components. This group consists of several mechanical components. Two are rather simple, pure inertia and ideal gear and are often used in connection with servo systems. Furthermore the following types of non-ideal gears are to be modelled, planetary gearbox, planetary gearbox with synchronous belt and harmonic gearbox. [19]. The element for the flexible robot arm is described in more detail.

Flexible link. When setting up a mathematical model of the flexible link there exist several different methods to obtain a reliable model. The model that is used in this work is stated in the following but a more detailed description of the mathematical modelling can be found in [17].

The partial differential equation describing the flexibility of the light weight link is

$$EI\frac{\partial^4 w}{\partial x^4} + \rho A\frac{d^2 w}{dt^2} = -\rho A\,(x+h)\,\ddot{\theta}_b \qquad (1)$$

where x represents the location on the link, h is hub distance (distance from rotation axis to the beginning of the flexible link), w is the flexible displacement from the corresponding rigid link motion (see figure 4), E is Youngs module, I is the area moment, ρ is the density, A is cross section area, $\ddot{\theta}_b$ is the angular acceleration of the output shaft of the actuator.

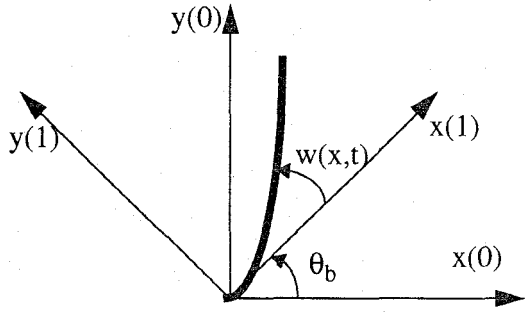

Figure 4. Flexible link definitions

The MSL model of the flexible link has the following inputs, rotation angle of the clamped end (rad) and angular acceleration of the clamped end (rad/sec) and the following outputs, absolute end point angle (rad), torque load (Nm), array of strain (m^{-1}) and resonance time functions. The MSL symbol and dialogue box are illustrated in figure 5.

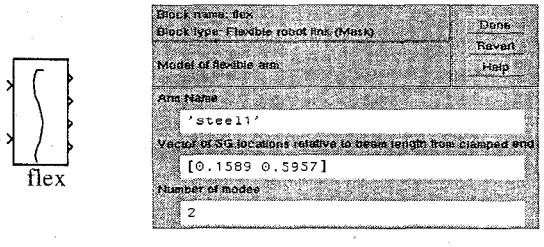

Figure 5. MSL symbol and dialogue box of the flexible link

The initialisation commands of the mask are

```
[Jb,alfa,Rw,Ww,Kmom,Phi2dd,PhiL]=
msldata('flexible_arm',@1,@2,@3)
```

where the involved variables has the following meaning

Jb	Moments inertia of rigid link with respect to the output shaft (Kgm^2)
alfa	$\ddot{\theta}_b$ to $\ddot{q}$ distribution vector.
Rw	Viscose damping matrix
Ww	Squared resonance frequency matrix (rad^2/sec^2)
Kmom	q to flexible torque feedback
Phi2dd	q to strain matrix
PhiL	Mode shape functions evaluated at x=L.

From the initialisation commands it can be seen that all parameters are read in the database file.

Generation of equations of motion. The dynamic equation of a robot is written in the standard form [4], [7]:

$$\tau = M(q)\,\ddot{q} + C(q,\dot{q})\,\dot{q} + F(\dot{q}) + G(q) \qquad (2)$$

Expressions for the matrix functions have to be derived and input into a block that contains a MATLAB function that implements the functions. There is a number of possibilities for deriving the matrix functions M, C, F, and G from knowledge of the construction of the robot.

- Derivation by hand. This is normally only possible for very simple systems and the chance of mistakes is rather large.
- Numeric calculation using the Robotics Toolbox [3]. The toolbox contains functions that from a description of the construction of the robot is able to derive numerical expressions for the matrix functions. The problem is that these matrix functions depends on the state of the robot and the numerical expressions has to be calculated at each time step.
- Symbolic calculations using general symbolic packages. [8] [21]. Using general symbolic packages like Mathematica or Maple V it is possible to derive symbolic expressions for the matrix functions that in turn can be imported in the MATLAB. There are specialised tools available for as for instance Robotica for Mathematica to solve this kind of problems.

  A more enhanced tool has been made for Maple V, which combines the use of Simulink graphical interface and the Symbolic Toolbox from MATLAB. This is constructed as a compagnion to MSL for modelling the mechanical part of a system. [20]. It utilizes the symbolic manipulation engine of Maple V symbolic algebra package accessible in MATLAB/Simulink environment in the form of Symbolic Math Tooolbox. The structure of the software architecture used in MAPLE generated Simulink C-mex function is shown in [20].
- Dymola [5] is a package specially well suited for solving this kind of problems. In [18], [16] and [20] it is

shown how simple it is to specify the construction of a robot and calculate the symbolic expressions for the matrix functions in MATLAB syntax.

- DADS (Dynamic Analysis and Design System) is a multi-body systems modelling package from CADSI, Inc. [13]. Recently it has been integrated with Simulink via DADS/Plant module implemented as an s-function. Using DADS/Plant graphical user interface models are built from joint and force elements. After defining the mechanical model DADS/Plant assembles the equations of motion and uses Simulink integrators to solve state equations of the overall system incorporating the mechanical part. When simulating the closed-loop system DADS/Plant receives the force and torque input signals from Simulink and returns velocities and accelerations.

Pneumatic blocks. A large number of mechatronic systems contains pneumatic elements. [1]. There are many advantages and only a few drawbacks of using pneumatic actuators in such applications. The drawbacks can be minimized by the special actuator design and now, pneumatic actuator are more widely applied for robotic systems, especially as drives for gripper systems, and drives for fine positioning systems. In MSL a number of pneumatic elements have been modelled which makes it easier to design robotic systems with pneumatic parts. The general structure of pneumatic systems incorporates following main elements: electrical part, mechanical part, valve, actuator (cylinder or bellow). The most important are the valve and actuator because these two components introduce into the system the main nonlinearities. There are strong interactions between all components of the system. The input section of the model consist of PWM type component. The library of pneumatic blocks includes the following elements:

- PWM element
- actuators (metal bellows and cylinder),
- valves (one-stage jet-pipe and two-stages Moog),
- flow meter system,
- mechanical part of positioning system.

5 Conclusion

In this paper the concept and design considerations for the Mechatronic Simulink Library are given, along with a describtion of the prototype implementation. MSL is a component library and a methodology for modelling mechatronics systems. The library and methodology has been used to model several mechatronics objects.

6 Acknowledgement

The support by the CEC under the COPERNICUS project CP93: 10119 'Dynamic Control of Robotic Manipulators - mechatronics approach' is gratefully acknowledged.

7 References

[1] Maurice Bétemps. Use of mechatronic concept for the realization of a feedback control actuator with metal bellows. some applications. In *2nd Japan-France Congress on Mechatronics*, Takamatsu, Japan, Nov 1994. Invited paper.

[2] F. E. Cellier. *Continuous System Modelling*. Springer-Verlag, New York, 1991.

[3] Peter I. Corke. *Robotics Toolbox*. CSIRO, 1994.

[4] John J Craig. *Introduction to Robotics, Mechanics amd Control*. Addison-Wesley, second edition, 1989.

[5] H. Elmqvist. *Dymola – Dynamic modelling Language, Users Manual*. Dynasim AB, 1994.

[6] L. Ljung. *System Identification: Theory for the User*. Prentice Hall, 1987.

[7] P. J. McKerrow. *Introduction to Robotics*. Addison-Wesley, 1991.

[8] John Nethery. *Robotica, User's guide and reference manual*, 1993.

[9] Anders Pjetursson and Ole Ravn. *Mechatronic Simulink Library, Reference Manual*. IAU, DTU, 2. ed, Oct 1995.

[10] Anders Pjetursson and Ole Ravn. *Mechatronic Simulink Library, Users Guide*. IAU, DTU, 2. edition, October 1995.

[11] Anders Pjetursson and Ole Ravn. System description of 3 DOF rigid robot. Technical report, IAU, DTU, 1995.

[12] Anders Pjetursson and Ole Ravn. System description of flexible robot arm. Technical report, IAU, DTU, 1995.

[13] W. Prescott. DADS/Plant nonlinear mechanism modelling for control design. In *MATLAB News and Notes*, pages 10–11. MathWorks, 1995.

[14] Ole Ravn and Anders Pjetursson. Computer aided control engineering: Process design models and tools. Technical report, IAU, DTU, 1995.

[15] Ole Ravn, Anders Pjetursson, and Anders Mortensen. Modelling and simulation of mechatronic systems. Technical report, IAU, DTU, 1994.

[16] Ole Ravn and Maciej Szymkat. Mechatronic approach to robotic modelling - software engineering perspective. In *MMAR'95*, Miedzyzdroje, Poland, September 1995.

[17] Morten Rostgaard. *Modelling, estimation and Control of fast Sampled dynamical Systems*. PhD thesis, IMM, DTU, 1995.

[18] Maciej Szymkat, Ole Ravn, Andrzej Turnau, Krzysztof Kolek, and Anders Pjetursson. Integrated mechatronic modelling environments. In *ICRAM'95*, vol. II, Istanbul, Turkey, August 1995.

[19] Tadeusz Uhl, Wojciech Lisowski, Tomasz Bojko, and Janusz Ród. Identification of robotic joints models using model adjustment technics. In MMAR'95, Miedzyzdroje, Poland, September 1995.

[20] Tadeusz Uhl and Janusz Ród. Modelling and identification of robotic systems using MATLAB/SIMULINK environment. In *ICRAM'95*, Istanbul, Turkey, August 1995.

[21] Stephen Wolfram. *Mathematica, a system for doing mathematics by computer*. Addison Wesley, 1991.